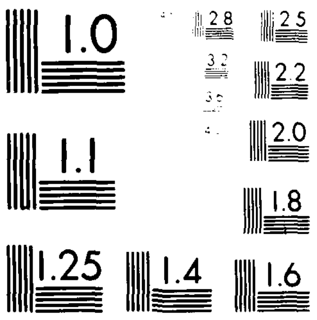
AD-A109 507 ARMY ELECTRONICS RESEARCH AND DEVELOPMENT COMMAND FO--ETC F/G 9/3
A HIERARCHICAL GRAPHIC SOFTWARE PACKAGE FOR DOCUMENTING CIRCUIT--ETC(U)
NOV 81 H RIEBSAMEN
UNCLASSIFIED DELET-TR-81-20 NL

1 OF 1
AD A
109507

END
DATE
FILMED
02-82
DTIC


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

LEVEL II

12

AD A109507

RESEARCH AND DEVELOPMENT TECHNICAL REPORT

DELET-TR-81-20

A HIERARCHICAL GRAPHIC SOFTWARE PACKAGE FOR DOCUMENTING CIRCUIT DESIGN LAYOUT

HENRY RIEBSAMEN
ELECTRONICS TECHNOLOGY & DEVICES LABORATORY

NOVEMBER 1981

ERADCOM

US ARMY ELECTRONICS RESEARCH & DEVELOPMENT COMMAND
FORT MONMOUTH, NEW JERSEY 07703

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER DELET-TR-81-20	2. GOVT ACCESSION NO. AD-A109 507	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* A Hierarchical Graphic Software Package For Documenting Circuit Design Layout.		5. TYPE OF REPORT & PERIOD COVERED Final
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Henry Riebsamen		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Microelectronics Division Electronic Technology and Devices Laboratory ERADCOM, Fort Monmouth, N. J. 07703 DELET-IC-K		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 1L1 62705 AH94 05 11 13
11. CONTROLLING OFFICE NAME AND ADDRESS US Army Electronics Research & Development Command, Fort Monmouth, N. J. 07703 (ERADCOM DELET-IC-K		12. REPORT DATE November 1981
		13. NUMBER OF PAGES 35
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Hierarchical Graphics Software

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This paper describes the processing techniques and characteristics of a graphic documentation software package which was developed for plotting hierarchically-modular circuit design layouts using a dot-matrix printer/plotter. This software can be used for other graphic applications as well so long as the data is supplied in the required descriptive language format. This software package will process modularly partitioned circuits or structures to any nested level and allow accumulative rotation and mirroring

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

(B orientations) of the modules regardless of nesting level. The software uses design data in RCA's Design File Language format ~~(see Appendix A)~~ and accesses data for the modules from library files using searches on name. If the graphic presentations require more than one page width, multiple segments (pages) are spooled out to the printer/plotter with fiducial reference marks for joining the segments together. Complete integrated circuit layouts may be plotted using either cell outlines or complete cell details. Other options include scaling, rotation, windowing, and shading of polygons. A single mask level may be selected for plotting, if desired.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

A. General

Background

More efficient design automation processes are needed to satisfy the demand for larger and more complex circuits. Hierarchically modular processing of system or circuit design algorithms can lead to improved computer processing efficiency as well as the minimization of design data storage requirements. The use of standardized modules, especially if used repetitively (multiple instances) in hierarchically modular design structures, can contribute greatly to the attainment of the desired efficiencies. The alternative, as used currently by many algorithms, is to fracture the circuit design data to primitive functions or geometries (smashing) prior to processing. To do otherwise requires a memory stack oriented recursive process. When, as required for some module placement and layout plotting operations, scaling, rotation, or mirroring of modules are not necessary, the problem of accumulative effects is introduced over the range of structured design levels.

Hierarchically-modular partitioning of a circuit is illustrated in Figure 1 for a structure of three levels. In this example, the overall design is a chip (level 1) which is partitioned into a group of components with conductor connectivity specified so that their composite behavior satisfies the behavioral specification of the chip. These components (level 2) are a group of macrocells (modules requiring further partitioning) and cells (modules not requiring further sub-division). Each macrocell is then partitioned, in a like manner, to result in a group of cells (level 3) with the required conductor connectivity specified.

Areas of investigation.

Hierarchically modular processing can be useful in numerous areas of design automation. For example, in the integrated circuit design process, hierarchically modular techniques can be used for multi-level simulation, module placement and conductor routing, design rule and connectivity checking, and documentation. The need to investigate multi-level design processing techniques for hierarchically structured circuits prompted development of the subject graphic documentation facility, which uses a dot-matrix printer/plotter. This paper describes the graphic utility and the techniques used that are relevant to hierarchically modular processing.

B. Graphics Requirements, Constraints, and Options.

Software requirements.

Several requirements were established and desirable attributes considered for the graphic software before proceeding with development.

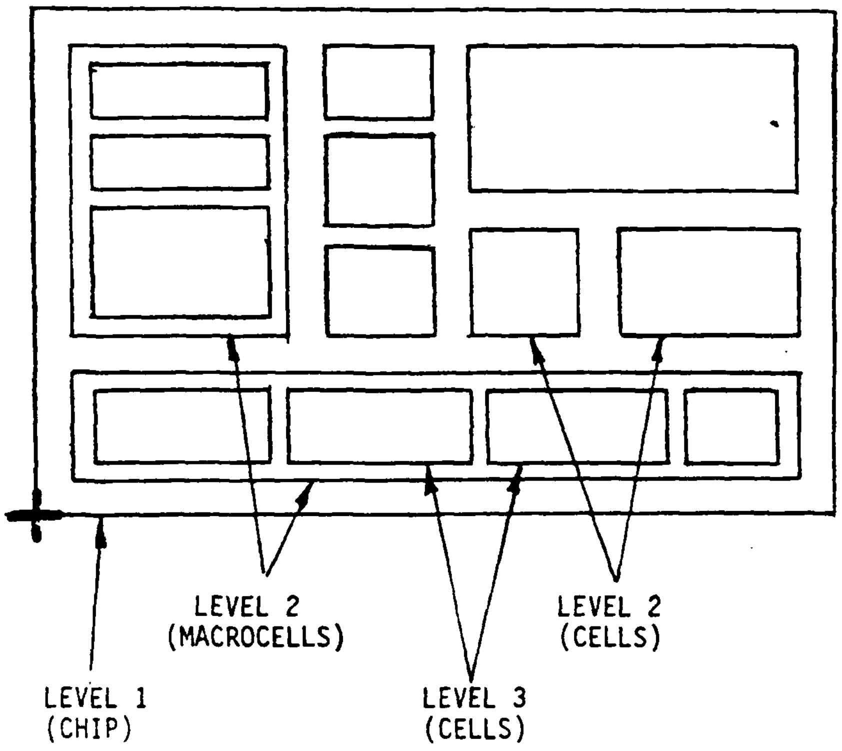


FIGURE 1. MODULE PLACEMENT DIAGRAM FOR A HIERARCHICALLY PARTITIONED CIRCUIT.

The following software requirements were adopted:

The software shall be able to make use of integrated circuit design files and cell libraries from the Army's current Computer Aided Design and Design Automation System (CADDA).

The software shall be able to handle circuit designs that have been modularly partitioned in a hierarchical manner without reducing the data to primitive geometries (smashing).

A desirable feature to be considered for incorporation into the software was a shading (polygon fill) technique which would be:

- Capable of overlaying shading patterns.
- Adaptable for use with a color printer/plotter (such as supplied by Trilog, Irvine, California)
- Adaptable for generating data for mask generation without reducing the design data to a single modular level (smashing).

Design File Language (DFL)

Because of its modular nesting capability, RCA's Design File Language (See Appendix A) was selected as the preferred of the two output design data formats available from the Army's current CADDA system.

Geometric Constraints.

The DFL language, and thus the subject graphics program, limits the types of geometric figures to: general polygons, ortho-polygons, polygons that are holes in other polygons, and paths that consist of connected line segments.

For the processing procedures used in this software, the following constraints must be applied to the geometric figures:

- Line segments and the distances separating line segments, when transformed for plotting, must be no shorter than twice the pitch of the dot spacing of the printer/plotter output.
- Adjacent line segments for polygons and paths should not form angles that are less than forty-five degrees. That is, sharp points must be chopped off.

Plotting Data and Options.

An interactive option editing program is used to initialize, change, list, and save the data and program options required by the graphic software. These options include:

o Dimension system (English, Metric, or Integer).

o Overall dimensions and border size.

  The minimum and maximum coordinate values of the x and y ranges of the design layout are required.

o Window size

  If the user elects to plot the complete design, the window size is defaulted to the overall dimensions; or else the minimum and maximum coordinate values for the selected area of the design are required.

o Resolution (units per dot-spacing).

o Scale factor

  A scale factor may be specified by the user. If not selected, or the selected value is in conflict with the resolution specified, the scale factor is defaulted to a value that is a function of the resolution. The scale factor may be optimized so that the plots fill a page width or number of page widths.

o Polygon outlines.

  If the polygon outline option is selected, outlines of polygons will be drawn.

o Shading of Polygons.

  A polygon shading option may be selected to enhance the presentation of design layout. For integrated circuit design layout, this feature associates a different shading pattern to each mask level of the fabrication process.

o Character option (EBCDIC or ASCII).

  The graphic software was written to use ASCII character code to conform to the CADDA facility. However, an EBCDIC to ASCII converter was provided since design data output from the current design system has data keys coded in EBCDIC.

- o Plot mode.

  The plot mode option enables the user to plot a selected mask level or all mask levels.

- o Cell outline option.

  Module outlines or complete details can be used for the graphic presentation by referencing the desired library.

Modular Nesting Options.

Associated with each design module are the following options:

- o Rotation and Mirroring of Library Elements.

  Each library call in a design file includes a parameter to indicate which of eight allowable orientations is to be applied for the call. See Appendix A for orientation detail.

- o Scaling of Library Elements.

  Each library call in the design data may include a scaling parameter in either or both the x and y-coordinate directions.

C. Processing Procedures

Design File Ordering.

In an effort to minimize library searching time, library calls in the design file are ordered by name, using a 3 digit collating routine.

General Program Structure.

The general program structure consists of a polling routine (See figure 2) for directing the processing as a function of the keys (See Appendix A) in the design data file. Data processing continues on each modular nesting level of design structure until a library (Q) call increments the nesting level or an end of library definition (Z) call returns processing to the next higher level of design structure.

Parameters that are a function of nesting level are stored in arrays used as stacks so that these parameters are available when returning from processing a module call to the next higher design structure level. These parameters include accumulative rotation, accumulative scale factor, accumulative fiducial reference dimensions, mask level, line width, and module name. Similarly, a series of consecutively numbered files are used as a file stack to store the data for the modules being processed. These arrays and files are over-written as needed in traversing the levels of a hierarchically structured design. Figure 3 is a flow diagram of the library searching, copying, and module processing initialization routine.

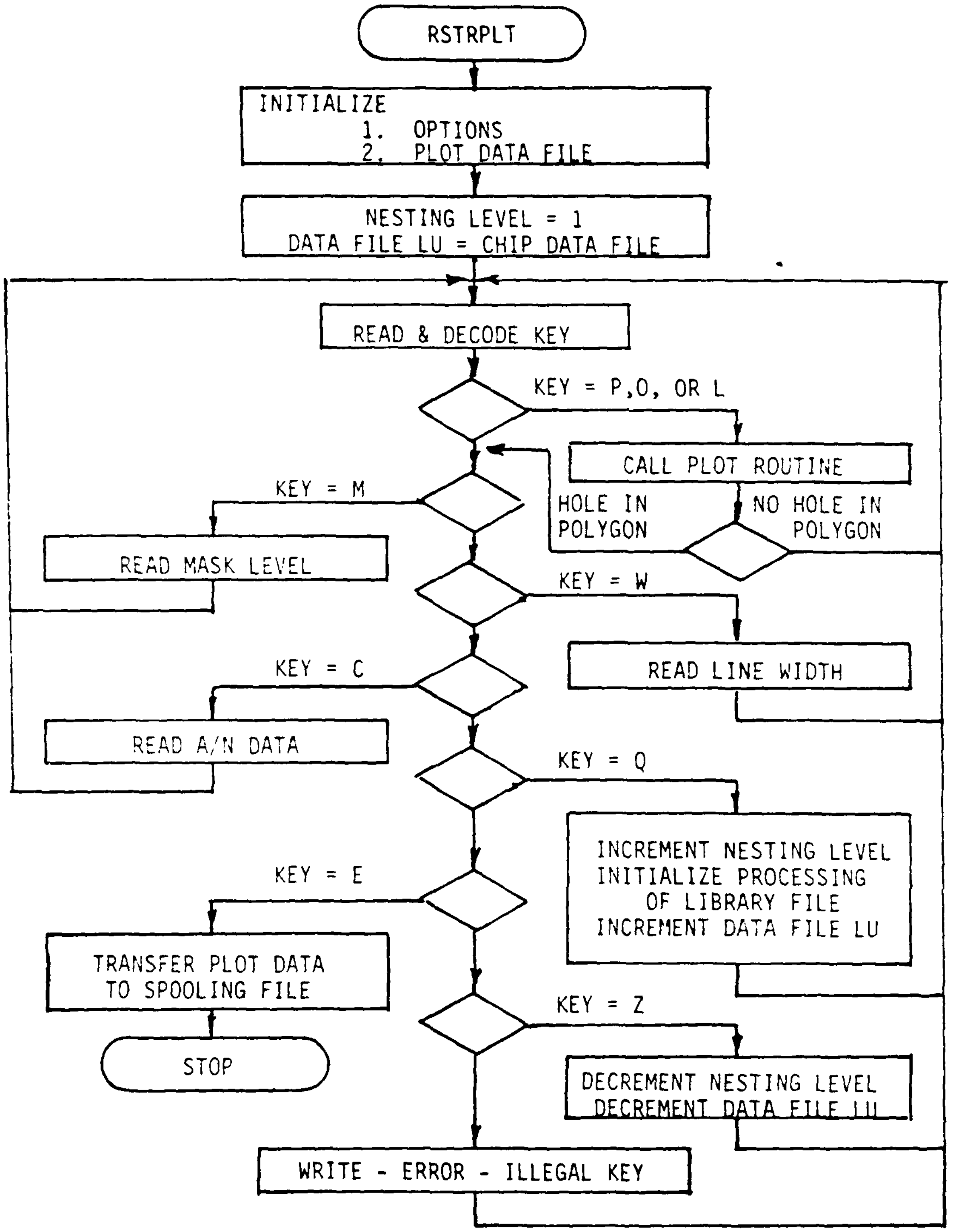


FIGURE 2. DESIGN FILE KEY POLLING ROUTINE

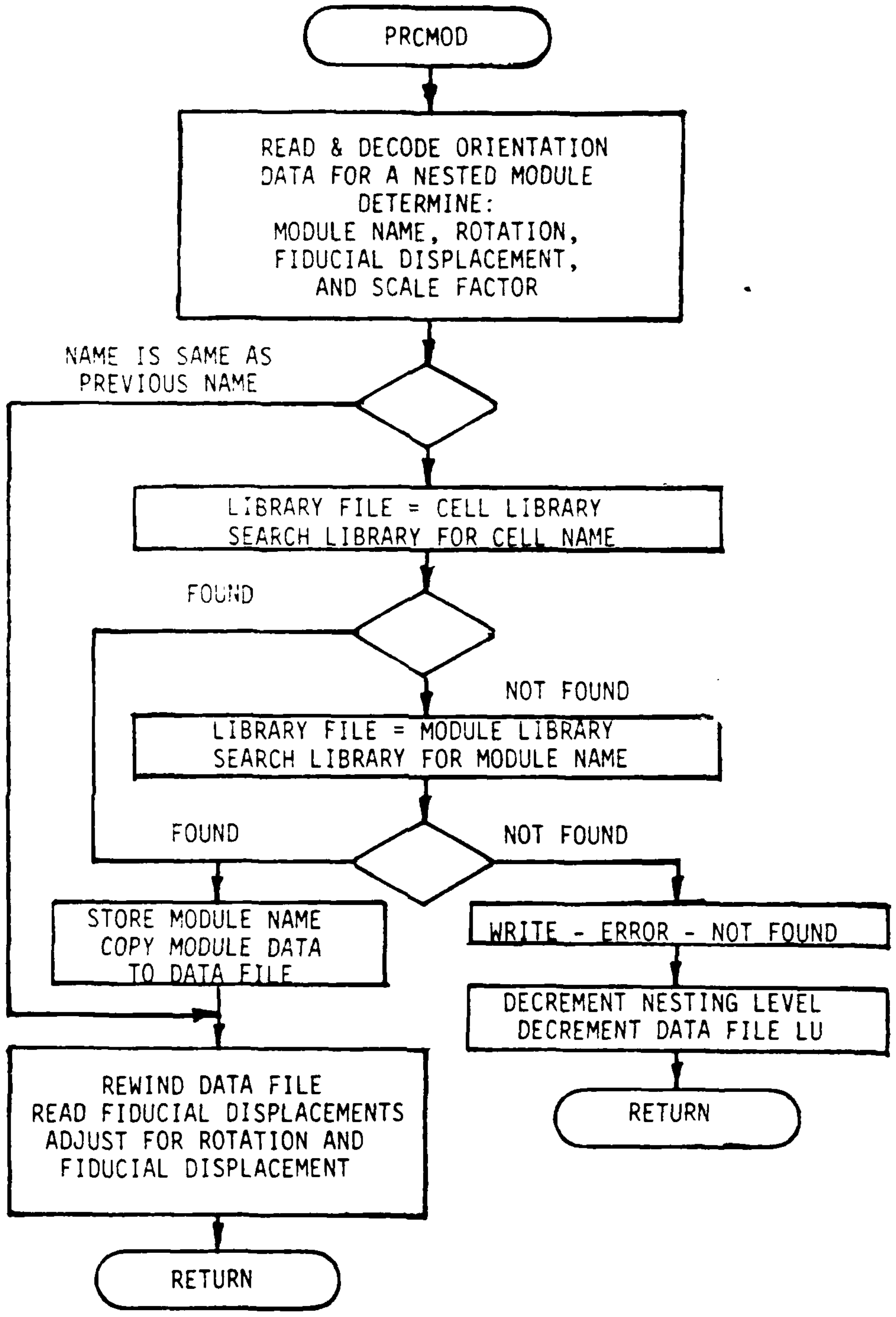

PRCMOD
READ & DECODE ORIENTATION DATA FOR A NESTED MODULE DETERMINE: MODULE NAME, ROTATION, FIDUCIAL DISPLACEMENT, AND SCALE FACTOR
NAME IS SAME AS PREVIOUS NAME
LIBRARY FILE = CELL LIBRARY SEARCH LIBRARY FOR CELL NAME
FOUND
NOT FOUND
LIBRARY FILE = MODULE LIBRARY SEARCH LIBRARY FOR MODULE NAME
FOUND
NOT FOUND
STORE MODULE NAME COPY MODULE DATA TO DATA FILE
WRITE - ERROR - NOT FOUND
DECREMENT NESTING LEVEL DECREMENT DATA FILE LU
RETURN
REWIND DATA FILE READ FIDUCIAL DISPLACEMENTS ADJUST FOR ROTATION AND FIDUCIAL DISPLACEMENT
RETURN

FIGURE 3. NESTED MODULE PROCESSING ROUTINE.

The Plot Data Files.

For the Printronix dot-matrix printer/plotter, a 780 X 780 dot-matrix picture format requires approximately 100,000 bytes of memory. If this size is tripled in both coordinate directions, close to a megabyte of memory is required for the picture matrix. Since these sizes are too large for array storage in random access memory, a random disc file is used for the plot data. The plot file is initialized to a12 blanks and then accessed randomly to modify the plot data as it is processed. I/O accesses use a 36 row by 67 byte buffer in order to minimize the number of I/O operations.

After all the plot data has been processed, the data from the randomly accessed file is converted to the form required for spooling to the Printer/Plotter.

Rotation and Mirroring of Library Elements.

Accumulative values for rotation and fiducial displacements are determined and stored in stack arrays at the corresponding level for each library call.

The accumulative rotation is determined for each library call by using a look-up table (figure 4) iteratively through each modular nesting level using the rotation symbols for adjacent levels to index into a table to obtain the accumulative effect. The accumulative rotation value is used to adjust the coordinate values for plotting. It is used as an index for vectoring to the proper coordinate adjusting routine. See Figure 5 for a diagram of the adjusting routine and Figure 11 for a demonstration of the accumulative rotation and mirroring capability.

To determine the accumulative fiducial displacement, the displacement in the library call is adjusted using the accumulative rotation at the next lower design level and then added to the fiducial displacement value at the next lower level of the design structure.

Coordinate Data Scaling

The origin, for drawings to be plotted, is located in the upper right-hand corner. Coordinate data for the vertices of polygons or paths are adjusted according to the user selected scale factor and windowing dimensions.

The adjusted y-coordinate values (PLOTY) correspond to the dot-row location in the plot matrix. See Figure 6a for the dimensional derivation. The adjusted x-coordinate values (PLOTX) correspond to the dot-spacing computed from a fiducial reference line with a displacement (DISPL) that is a function of the page number for drawings requiring more than one page width. See Figure 6b for the dimensional interrelationships.

SEE APPENDIX A FOR AN EXPLANATION OF ROTATION CODES

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	0	6	7	5	4
2	2	3	0	1	5	4	7	6
3	3	0	1	2	7	6	4	5
4	4	7	5	6	0	2	3	1
5	5	6	4	7	2	0	1	3
6	6	4	7	5	1	3	0	2
7	7	5	6	4	3	1	2	0

ROTATION INDEX FOR CURRENT NESTED LEVEL

ROTATION INDEX FOR NEXT LOWER NESTED LEVEL

FIGURE 4. LOOK-UP TABLE FOR ACCUMULATIVE ROTATION EFFECTS.

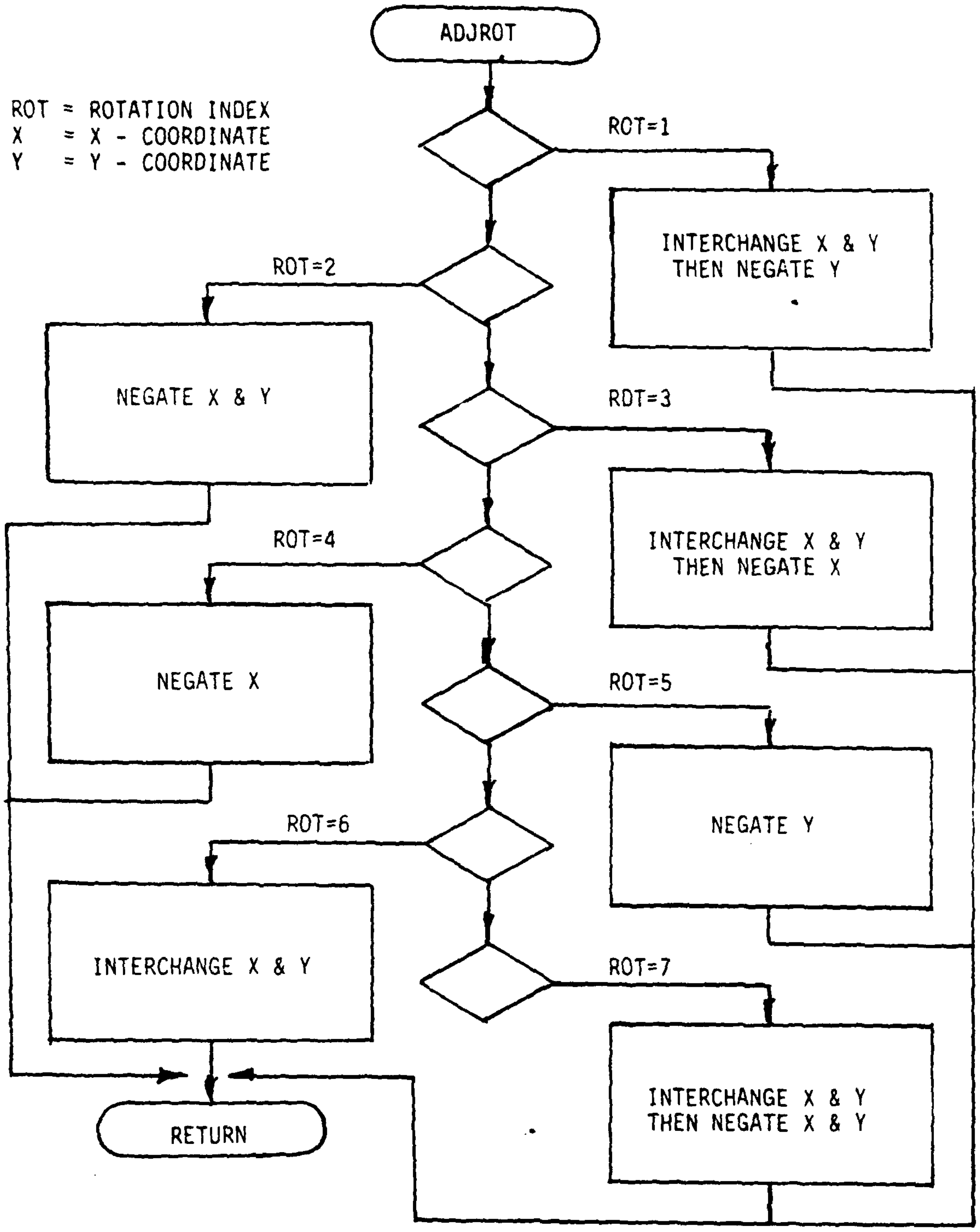
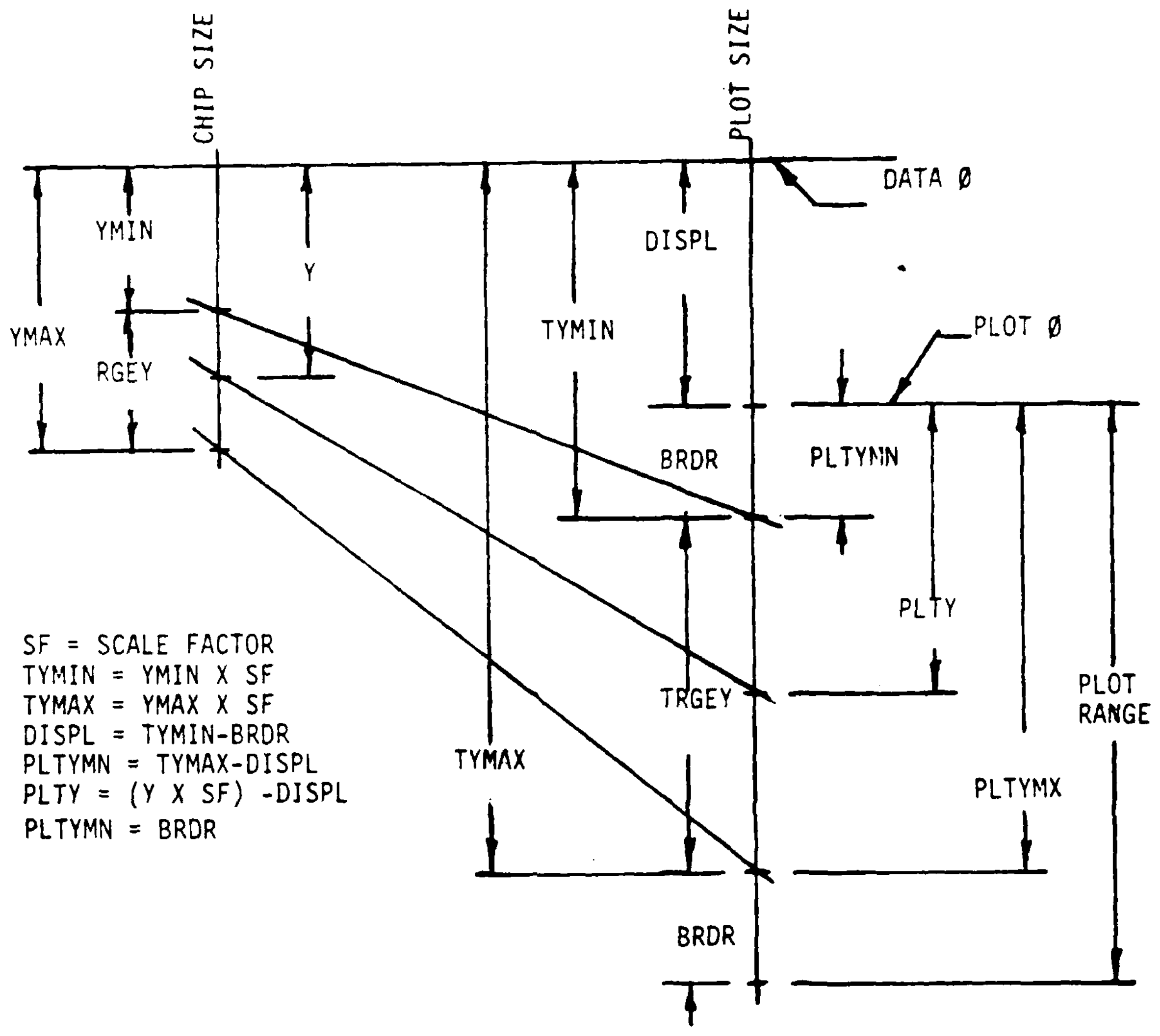
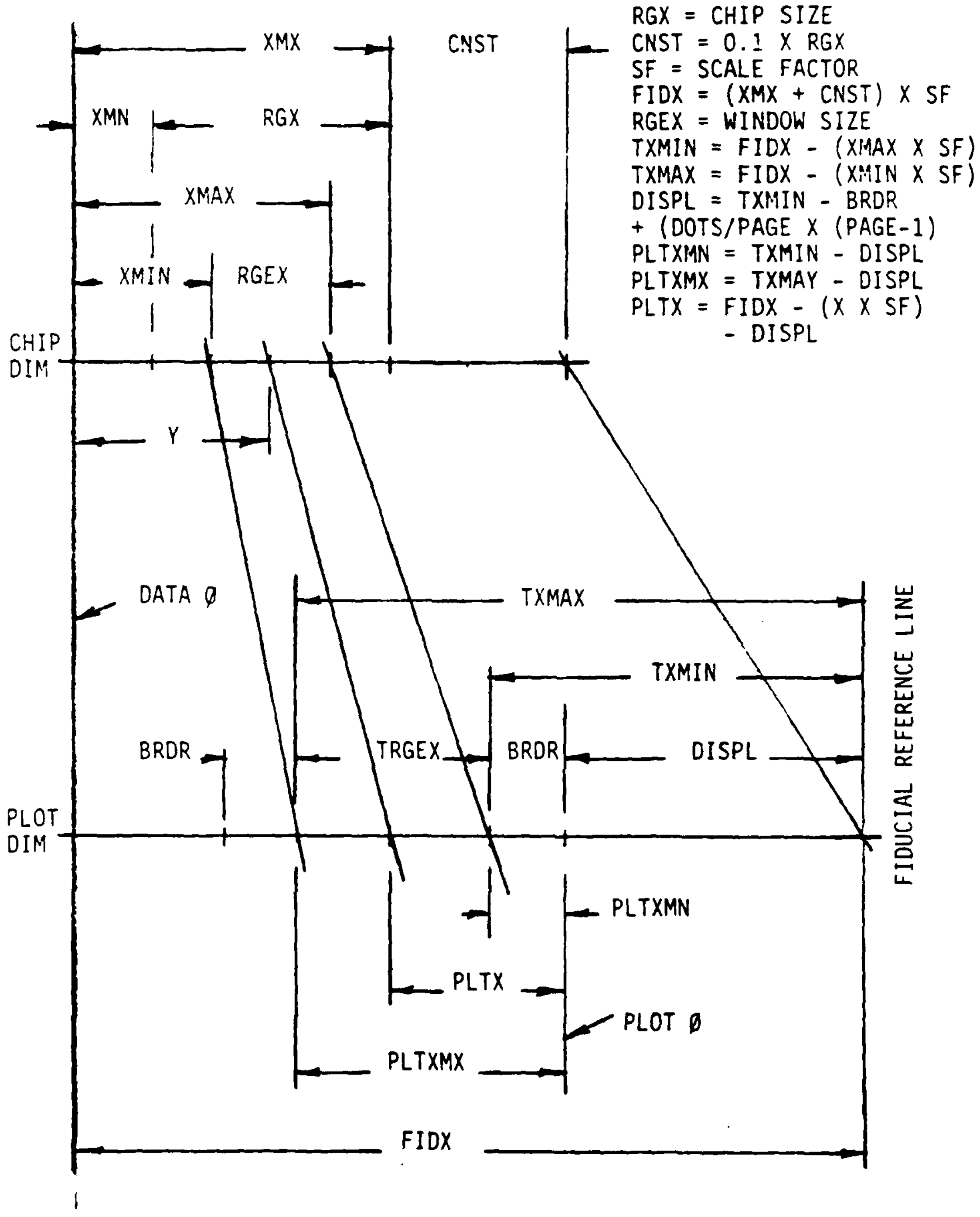


FIGURE 5. COORDINATE ADJUSTING ROUTINE FOR ROTATIONAL ORIENTATIONS.



SF = SCALE FACTOR
TYMIN = YMIN X SF
TYMAX = YMAX X SF
DISPL = TYMIN-BRDR
PLTYMN = TYMAX-DISPL
PLTY = (Y X SF) -DISPL
PLTYMN = BRDR

FIGURE 6a. Y - COORDINATE DIMENSION SCALING DIAGRAM.



RGX = CHIP SIZE
CNST = 0.1 X RGX
SF = SCALE FACTOR
FIDX = (XMX + CNST) X SF
RGEX = WINDOW SIZE
TXMIN = FIDX - (XMAX X SF)
TXMAX = FIDX - (XMIN X SF)
DISPL = TXMIN - BRDR
+ (DOTS/PAGE X (PAGE-1)
PLTXMN = TXMIN - DISPL
PLTXMX = TXMAY - DISPL
PLTX = FIDX - (X X SF)
- DISPL

FIGURE 6b. X - COORDINATE DIMENSION SCALING DIAGRAM.

The Plotting Technique.

The plotting technique developed for this software enables:

- Plotting polygons and paths per DFL language specification (Appendix A) except that the constraints described in paragraph B must be observed.

- Shading of polygons with patterns as a function of fabrication mask level. The shading patterns have been selected to enable esthetically reasonable composite patterns for representing the common areas of polygons that overlap one another.

The plotting technique adopted for this software recognizes vertices of polygons or paths to be critical points requiring special treatment for plotting. The y-coordinate values of a polygon or path are sequenced to get a series of plotting intervals in the y-coordinate direction. Further, those y-coordinate intervals that would overlap records in the randomly accessed data file are subdivided. See Figure 7. Note that, the partitioning assures that all vertices appear in the first row of a y interval.

The vectors appearing in each y interval and the vector count are determined. Then, for each y interval (segment), the x-coordinate values to be plotted, for each vector in each segment, are determined for each row in the segment.

The x-coordinate values for each vector are processed according to the vector characteristics. Separate routines are used for:

- Horizontal lines.

- Vertical lines.

- Skew lines greater or equal to forty-five degrees.

- Skew lines less than forty-five degrees.

Note that dot matrix representation of skew lines that are less than forty-five degrees, result in multiple horizontal line segments.

For horizontal lines, only the endpoints are listed, but are flagged for recognition when plotting.

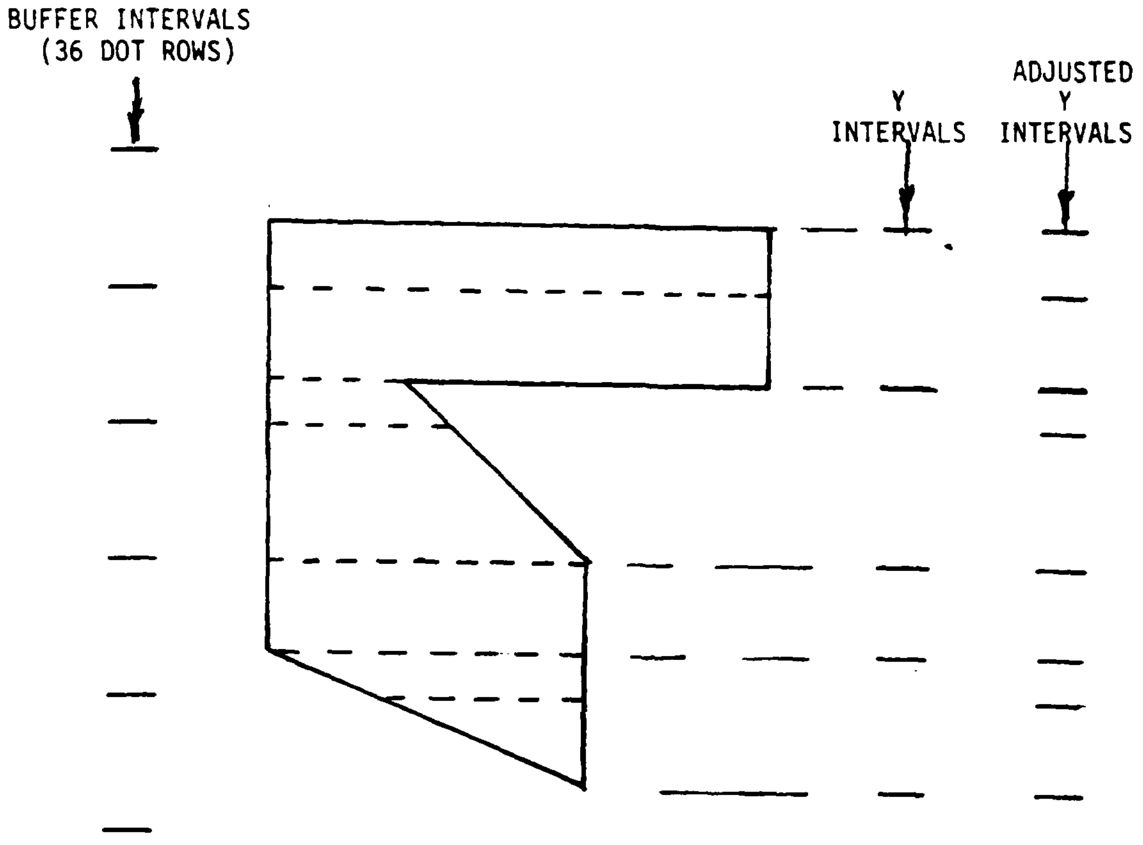


FIGURE 7. GEOMETRIC SEGMENTATION FOR PLOTTING.

The x-coordinate values in each dot row are then sequenced prior to the actual plotting or shading operation.

If flagged for line plotting, each dot row of x-coordinate values is scanned and the routines invoked for plotting points and/or horizontal line segments, as required.

Point Plotting (See glossary for logical and arithmetic definitions.)

The Printonix Printer/Plotter is byte oriented and requires bit seven of a data byte to be set to indicate that it is plot data rather than alpha-numeric data. With bit seven set, bits one to six correspond to the plot state of the next six dots on the row being processed. However, since data is processed more efficiently in half words (two bytes), the plot data is manipulated in this manner by the grapic software.

To plot a point on a given row, DIV 12 of the x-coordinate determines the word in the row to be modified and MOD 12 of the x-coordinate gives the position in the 12 dot interval to be modified. The MOD 12 value is used to index into a table to find the value to be OR'd with the word to be modified. Figure 8 pictures the look-up table plot representation for the twelve possible values.

Plotting a Horizontal Line.

To plot a horizontal line, MOD 12 and DIV 12 of the two x-coordinates are first determined. The DIV values give the initial and final word to be modified in scanning the row being processed. The initial MOD value is used to index into a table to get the value for the line segment of the first word to be modified. The final MOD value is used to index into another table to get the terminating line segment. The interim locations are modified by ORing with the code for a solid line of 12 dots. Figure 9 illustrates the look-up table plot representations for the initial and terminating segments of a horizontal line.

If the two DIV values are equal, ANDing the two values obtained using the MOD values will get the horizontal line segment representation within a single word of the dot row being processed.

Plotting lines that are other than horizontal.

Except for lines that form an angle of less than forty-five degrees with the horizontal, the x-coordinates are determined for each y-coordinate value using basic slope-intercept geometry. Each point is then plotted as outlined above.

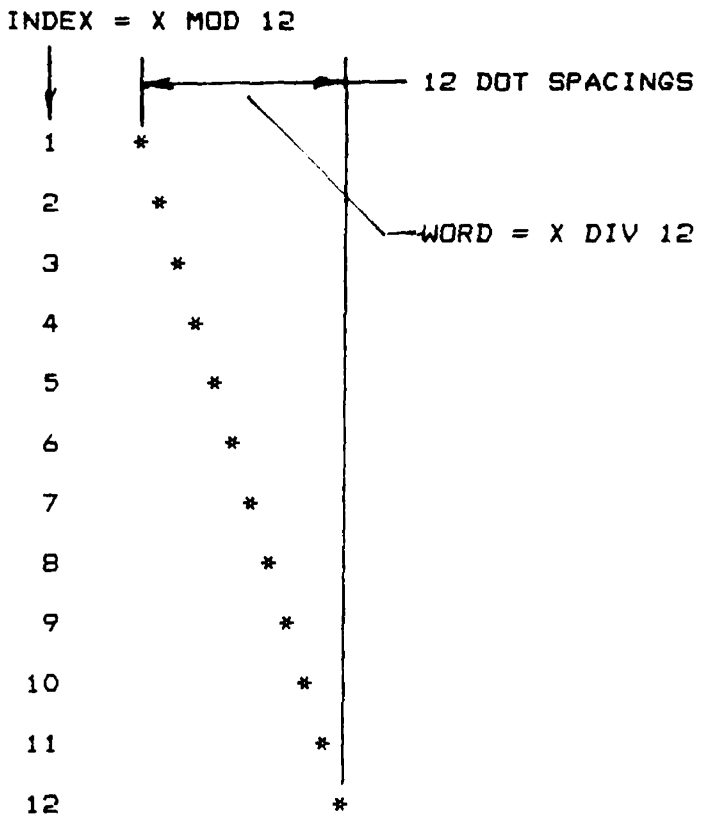


Figure 8. Look-up table representations for point plotting on a given row of plot matrix.

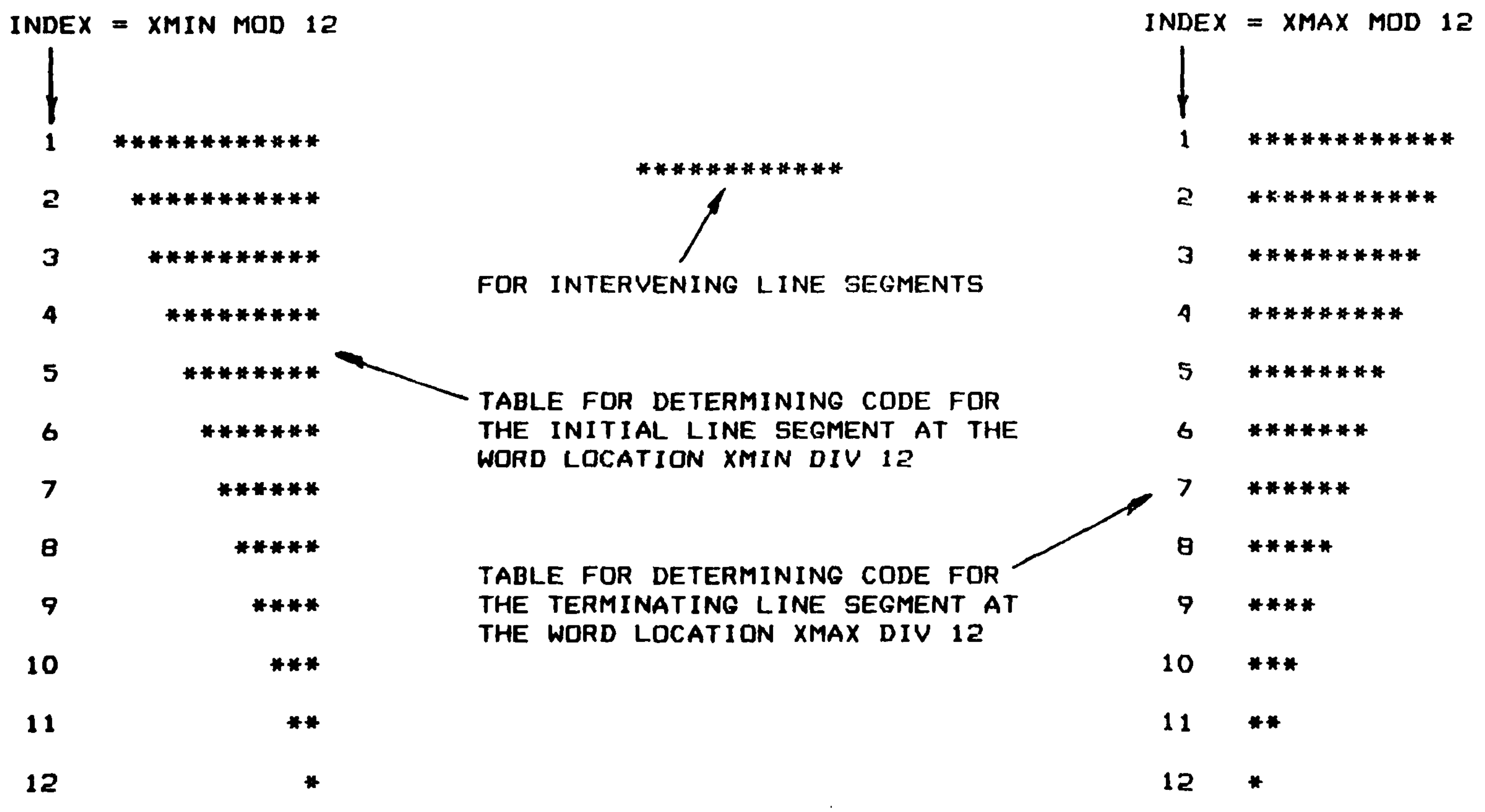


Figure 9. Look-up table representations for plotting horizdntal lines on a given dot-matrix row.

For those lines that form an angle less than forty-five degrees with the horizontal, the line is broken up into horizontal line segments and points which are then plotted individually as outlined above.

Shading.

If flagged for shading, each dot row of x-coordinate values for a polygon is scanned and directed to shade areas enclosed by the polygon except for areas that are holes in the polygon. For orthogonal polygons, the shading procedure is controlled by complementing a shading flag every time a line is crossed in horizontally scanning the picture field. However, for general polygons, the flagging procedure must be more sophisticated.

Each vector of a pair intersecting to form a vertex is flagged according to its orientation with a horizontal line through the vertex. Line segments are typed as horizontal (0), above the horizontal (+1), or below the horizontal (-1). Figure 10a illustrates the vertex and line orientations that are flagged for complementing the shading flag. Figure 10b shows the conditions for which complementation of the shading flag is to be ignored.

In shading a polygon, each interval flagged for shading on a dot row is processed by a call to the same routine used to plot a horizontal line, the only difference being the look-up tables used for obtaining the data used for modifying the horizontal interval. The shading values are obtained from look-up tables having an additional dimension since the shading patterns are a MOD function of the dot row being processed.

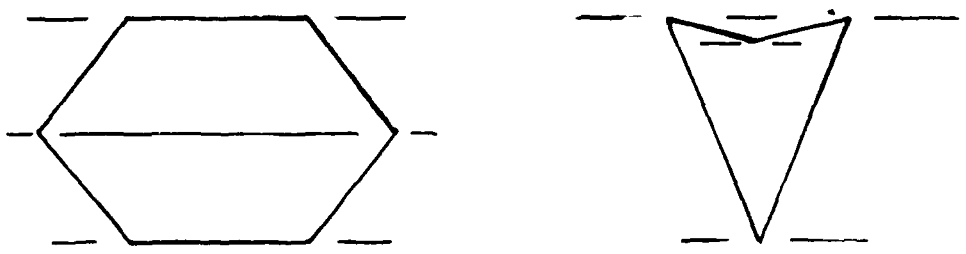
Note that the look-up tables for shading pattern data used for the Printonix dot-matrix printer/plotter are similar to those which would be required for the Trilog color printer/plotter.

D. Graphic Software Verification.

Development of the graphic software included systematic tests and demonstrations of the adopted options and capabilities. Some of the more important of these are as follows:

A graphic representation of a standard cell with:

- The polygon outline option.
- The shading option added to the polygon outline option.



a. Conditions for complementing

b. Conditions for not complementing

Figure 10. Polygon vertex criteria for complementing the shading flag.

The capability of plotting general polygons having sides at any angle within the geometric constraints outlined above for:

o The polygon outline option.

o The shading option added to the polygon outline option.

o Holes inside polygon outlines.

Library accessing for circuit designs which have been configured by the Army's MP2D placement and routing program for:

o The polygon outline option.

o The shading option added to the polygon outline option.

o The cell outline option using a cell outline library.

Multi-page graphic presentations.

Plotting selected areas of a design (windowing).

Library accessing for designs that have been modularly-partitioned in a hierarchical manner. This included accumulative rotation, scaling, and fiducial displacement to any number of nested levels.

See Figures 11 to 14 for examples of the plotting ıpabilities.

E. Development Results and Conclusions.

A hierarchical procedure was developed and demonstrated for plotting that does not require reducing the design data to primitive geometries (smashing) prior to processing. Its hierarchical capability is made possible by:

o A recursive technique which uses stacks for storing data that is a function of hierarchical nesting level.

o A table look-up technique that automatically adjusts for the accumulative effects of multiple rotations and mirroring.

A graphic capability has been developed and demonstrated for documentation of integrated circuits that have been processed by the Army's MP2D placement and routing program. Computer processing times for plotting a small chip varied from about five minutes for a small windowed area of the chip to about fourteen minutes for a complete plot using outlines for the cell representations. For complete cell details, appreciably longer time would be required.

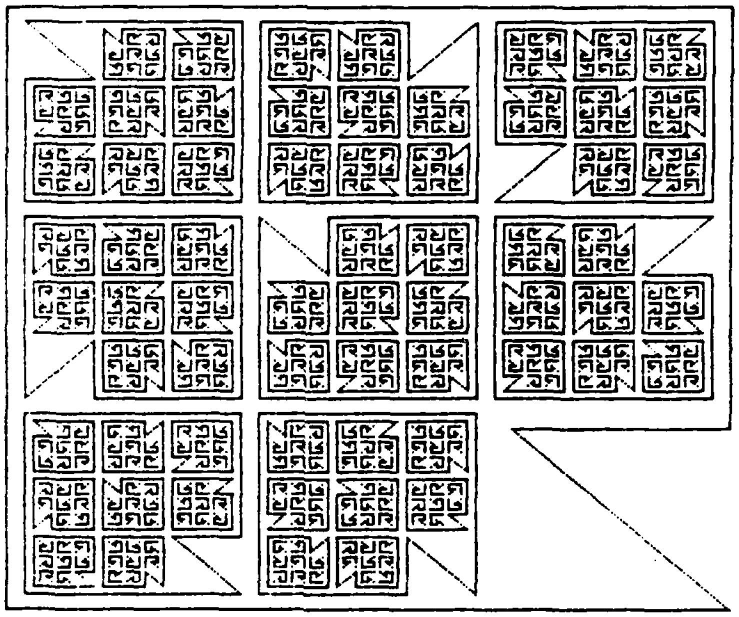


Figure 11. A test plot illustrating accumulative rotation and mirroring for four nested levels.

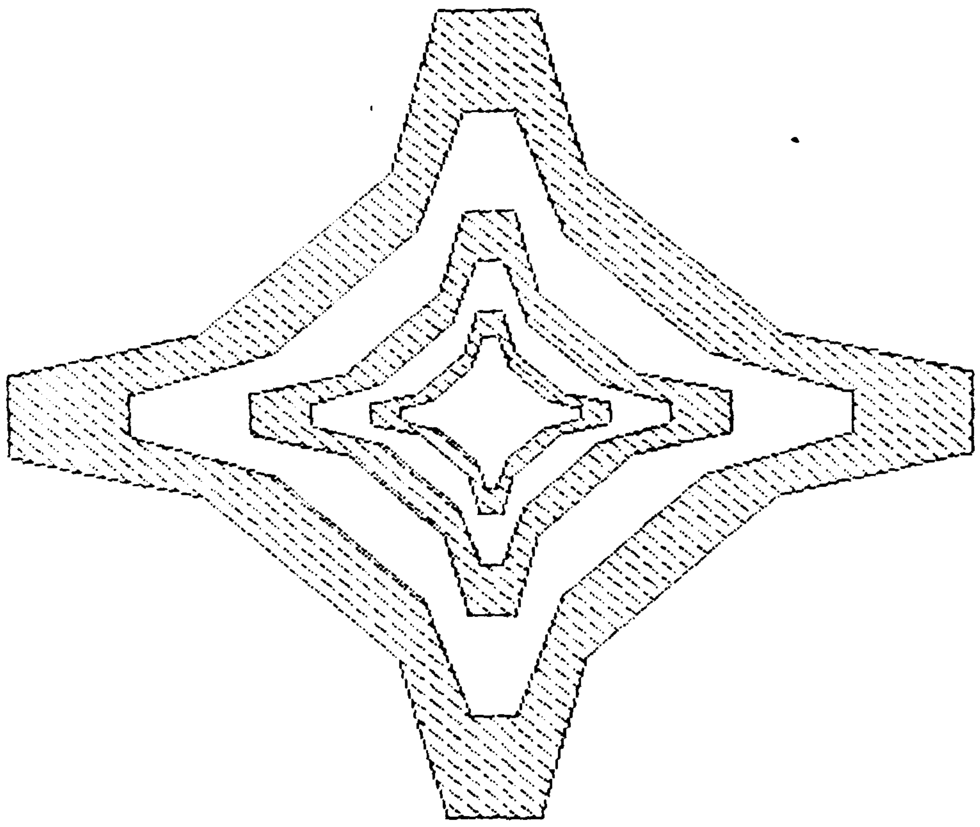


Figure 12. A test plot illustrating the graphics capabilities related to angles, scaling, shading, and holes in polygons.

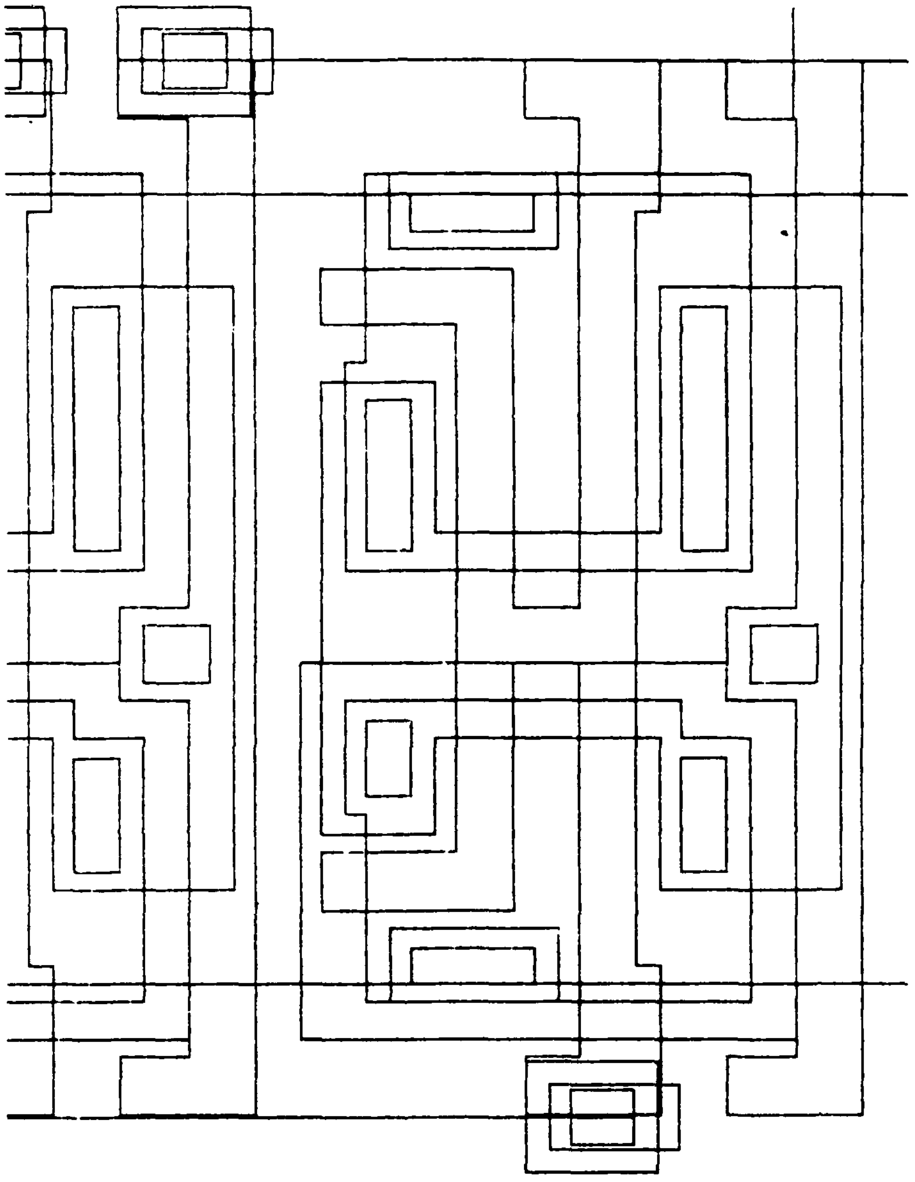


Figure 13. A test plot of a windowed area of an integrated circuit using the polygon outline option.

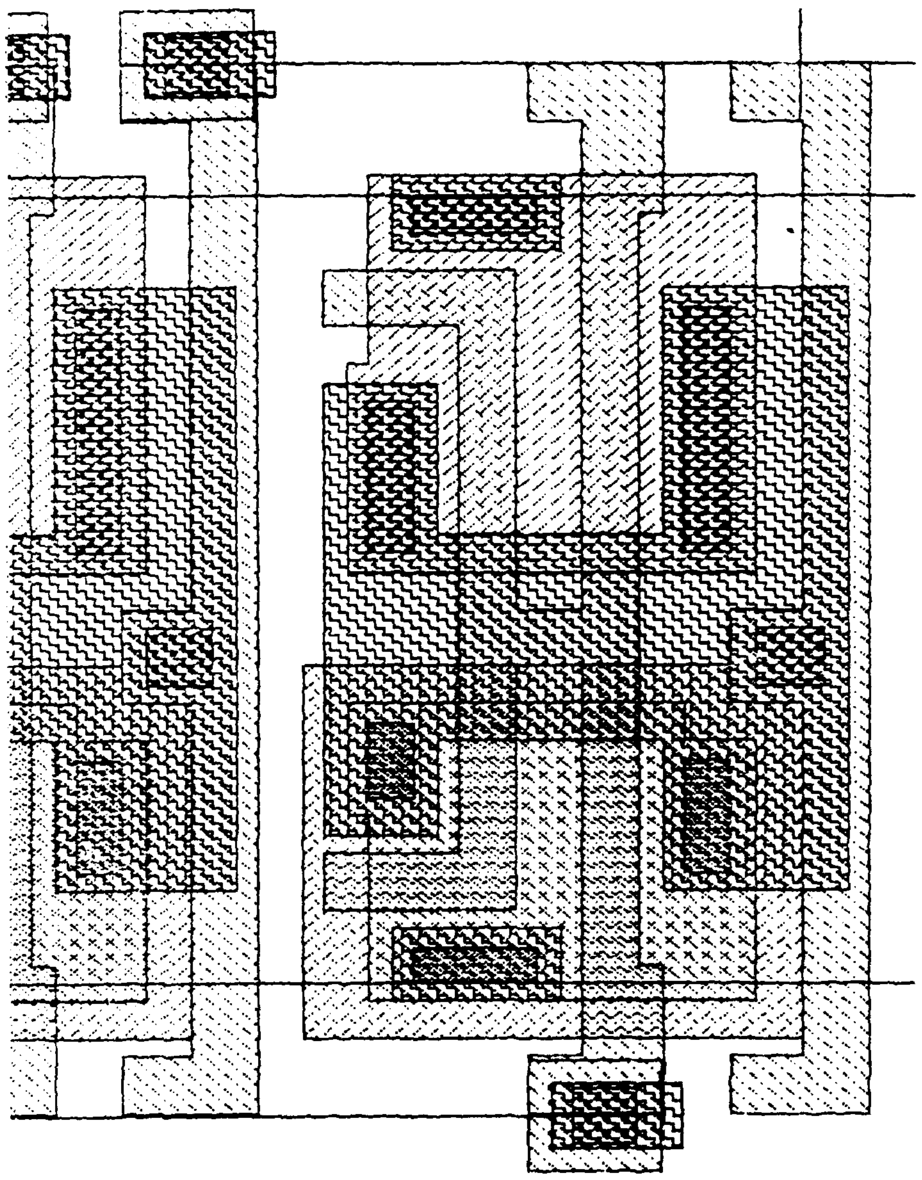


Figure 14. A test plot of a windowed area of an integrated circuit using the polygon outline and shading options.

A raster oriented shading technique was developed and demonstrated for use as a polygon fill operation in the subject graphic software. For integrated circuits, the shading pattern is selected as a function of the fabrication mask level. This technique is also considered suitable for:

a. Color graphics processing with the Trilog color printer/plotter.

b. Generating raster scan data for mask fabrication from data files of designs that have been hierarchically structured.

The graphic documentation package does not support text plotting at the present time. It was not introduced because the descriptive language used (DFL) does not provide this capability in the version used. It is needed to provide text and text position for cell identification, component number, and terminal numbers when outlines of cells are plotted instead of detail descriptions. The text data must be processed with proper rotation, but no mirrowing, for readability.

Although the graphic software package was designed primarily for plotting hierarchically modular circuit layouts, this graphic software package can be used for other applications. For example, with a simple design file editing program, it could be used for preparing illustrations for papers or reports.

Glossary

AND operator - The AND operator has the property that if P and Q are two integers in binary form, then (P AND Q) is a binary number for which each bit is 1 if both of the corresponding bits of P and Q are 1, else 0.

DIV operator - The DIV operator has the property that if P and N are two integers, then P DIV N is the truncated result of division of P by N.

MOD operator - the MOD (MODULO) operator has the property that if P and N are two integers then P MOD N is the remainder for an integer division of P by N.

OR operator - The OR operator has the property that if P and Q are two integers in binary form, then P OR Q is a binary number for which each bit is 1 if either or both of the corresponding bits of P and Q are 1, else 0.

Appendix A

DESIGN FILE LANGUAGE SPECIFICATION

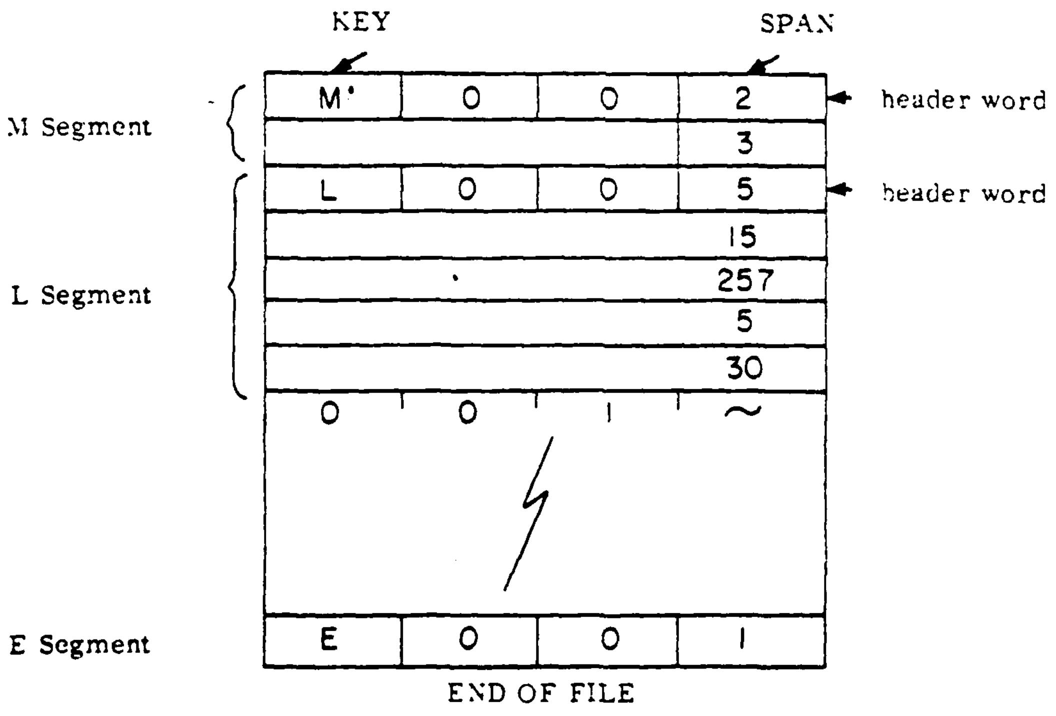
A. GENERAL SPECIFICATIONS

A Design File consists of segments, each of which begins on a word boundary and contains an integrated number of full words. The first word in each segment is called the Header.

The first byte (8 bits) of the header word is called the Key. The Key is an EBCDIC character which identifies the type of segment which follows. The fourth byte of the header contains the Span, which is the number of full words in the segment, including the header.

The significance of the remaining words in the segment depends on the type of segment, as determined by the key. The last segment in any file must be an E segment.

Sample Design File:



All coordinates in DFL Spectra 70 form are signed integers in units of 10^{-8} inch = 0.01 microinch. Thus the distance 1 mil is represented by the integer 100,000.

The maximum span of any segment is $255 = 2^8 - 1$. The maximum coordinate value is ±21.47483647 inches = $(2^{31} - 1)/10^8$.

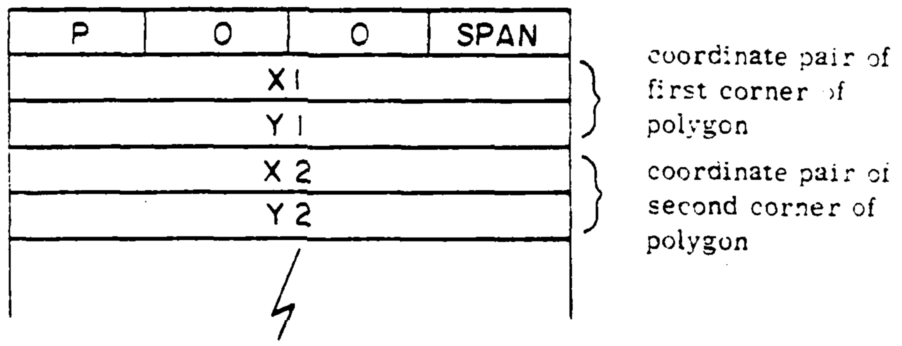
B. SEGMENT TYPES

1. Polygon Segment

Key: P

Span: 2N - 1, N = number of corners in polygon

Format:


P
0
0
SPAN
X1
Y1
X2
Y2
coordinate pair of first corner of polygon
coordinate pair of second corner of polygon

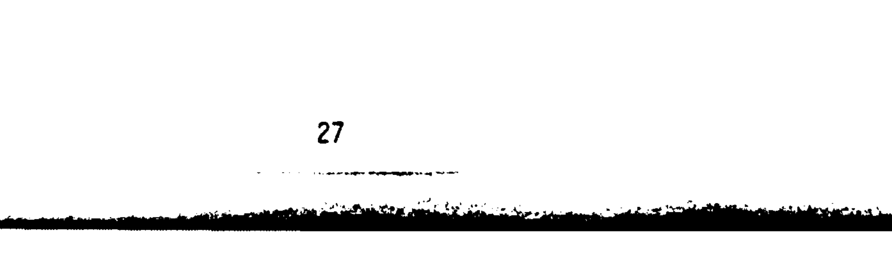
Interpretation: This segment specifies a closed polygon with a maximum of 127 corners. The sides may be at any angle with the x and y axes, but may not intersect each other. Closure is assumed from the last point in the segment back to the first point. This means the first point is not repeated at the end of the segment: a corner angle of 0° or ±180° is NOT permitted.

2. Orthogonal Polygon Segment

Key: 0

Span: N + 1, N = number of corners

Format:

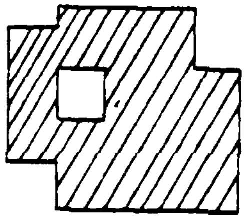

0
0
FLAG
SPAN
X1
Y1
X3
Y3
coordinate pair of first corner of 0 polygon
coordinate pair of third corner of 0 polygon

Interpretation: The 0 segment specifies a polygon all of whose sides are parallel to either the x or y axes. Each corner angle must be ±90°. Only the coordinates of every other corner (i.e., the first, third, fifth, etc.) are stored in the segment to conserve space. Sides may not intersect each other. The first side, starting at the first coordinate pair in the segment, must be parallel to the x-axis (horizontal). The maximum number of corners is 254. Closure is assumed as for the P polygon.

The FLAG identifies the orthogonal polygon as being either Normal, Exterior or Hole (see below).

FLAG	=	0	Normal
		1	Exterior
		2	Hole

It is often desired to specify a figure which is not simply connected; that is, it has an isolated area inside the figure which is not filled in (see sketch). An orthogonal polygon segment header contains a byte which can specify this condition and identify whether the particular polygon is an Exterior, or outside, polygon, or a Hole, specifying an empty space inside another polygon (see the definition of FLAG above).



Orthogonal Polygon

The following rules govern the use of OE (Exterior) and OH (Hole) polygons:

a. Only orthogonal polygons may form holes and exteriors.

b. Any OE may contain up to 200 OH polygons inside it (but no more than 800 corners total for all holes).

c. No OH may intersect another OH or the OE surrounding it.

d. Every OH must be inside an OE.

e. No OE may be inside an OH.

f. Exteriors may intersect; however, any holes in common will be filled.

g. Normal polygons may be placed inside holes.

Processing programs reduce an exterior and its holes down to a number of simply-connected normal polygons and fills them. Such a program will define an error if more than 200 sides are formed when this reduction is made.

3. Line Segment

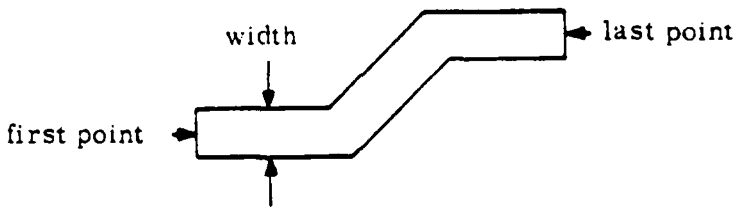
Key: L

Span: 2N-1, N = number of ends and corners of line

Format:

L	0	0	SPAN
X1			
Y1			
X2			
Y2			

Interpretation: A line is placed on a drawing by connecting the points in the line segment in the order in which they appear. The line will be of constant width as determined by the most recent width (W) segment. The ends are squared off at the first and last points, and each corner in the line is made continuous by intersecting the parallel sides of the two lines (except for the outside point of an acute angle, which is smoothed off). The line may intersect itself. The line may have no more than 127 corners, including end points.



4. Mask Segment

Key: M

Span: 2

Format:

M	0	0	SPAN
			LEVEL

Interpretation: The Mask segment specifies the mask level number on which any figures following the segment appear. The level specification holds true until the next mask segment changes it, or the file ends. The segment can specify any level from 0 to 255, but only levels 0-23 may be used by programs processing DFL.

5. Width Segment

Key: W

Span: 2

Format:

W	0	0	2
WIDTH			

Interpretation: This segment specifies the width of any figure specified by a line segment following this width segment. This width specification holds until the next W segment changes it, or the file ends. The value of W is an integer in units of 10^{-5} inch, and has the same limits as any other coordinate number.

6. End Segment

Key: E

Span: 1

Format:

E	0	0	1

Interpretation: The E segment marks the end of a design file, and is always the last segment in a file.

7. Comment Segment

Key: C

Span: Variable

Format:

C	FLAG	SPAN

Interpretation: The comment segment is defined to provide for future additional segment types. The specific interpretation of a given comment is indicated by the value of FLAG, which is arbitrarily assigned whenever the need arises. The span is set according to the conventional rules.

5. Definition Segment

Key: D

Span: 5

Format:

D	0	0	5
0	0	NAME	
x-fiducial			
y-fiducial			
-1			

Interpretation: The Definition Segment marks the beginning of a section of the design file which may be referenced elsewhere in the file by name. The NAME is a number between 1 and 32,767 = $2^{15}-1$ (considered a 2-byte signed integer). The x and y fiducials define a point which is used as a reference point when referring to the definition from elsewhere in the file. The coordinates are subject to the same numerical constraints as any other coordinate in DFL. The -1 (all 32 bits set) is the result of an historical accident.

The section of a design file intended to comprise the definition must be preceded by a D segment and terminated by a Z segment (see below). Once a definition is begun, it must be terminated with a Z segment before another definition is started. Two definitions by the same name may not exist in the same file.

Names are chosen by the user under the following constraints:

NAME	USE
1-199	User may choose at will. May be referenced only within the same file.
200-9999	Numbers assigned to users in blocks. User may use only those numbers assigned to him.
10,000 up	Reserved for system library use.

9. <u>End-of-Definition Segment</u>

Key: Z

Span: 1

Format:

Z	0	0	1

Interpretation: Terminates a definition, which began with a D-segment.

D-segment
All the segments in the definition
Z-segment

10. <u>Library Call Segment</u>

Key: Q

Span: 4, or 6 if scale factors are included.

Format:

Q	0	0	4 or 6
ROTATION	0	NAME	
x-fiducial			
y-fiducial			
x-scale			
y-scale			

Interpretation: A Q-segment in a file specifies that the definition whose number the Q-segment bears is to be placed in the position, with the rotation, with the scale factor given. The fiducial point of the definition will be placed at the fiducial point of the Q-segment (sometimes referred to as Q-call). All other points on the definition will be in the same position relative to the Q-call fiducial as they are within the definition itself.

The ROTATION byte specifies 1 of 8 possible orientations for the definition as it is laid in place:

ROTATION		POSITION
0		same orientation as in definition
1		90° clockwise

ROTATION	POSITION
2	180° clockwise
3	270° clockwise
4	Rotate about Y-axis
5	Rotate about X-axis
6	90° clockwise, then about X-axis
7	90° clockwise, then about Y-axis

The two scale factors are optional; if they appear in the Q-call, they specify that the definition is to be stretched or shrunk about the fiducial when placed on the artwork. The scale factor is a 32-bit number with binary point in the middle; i.e., a scale factor of 1 has the following bit pattern:

```
0000 0000 0000 0001 0000 0000 0000 0000
```

The largest scale factor is thus $2^{16}-1 = 65.535$ and the smallest is $2^{-16} \cong .000015$. Care must be taken that the product of the scale factor and the coordinates to which they are applied is neither too large nor too small for the purpose intended.

Since the X and Y scale factor may in general be different, care must be taken when applying both rotation and scale factors to a definition. The order is always:

a. Rotate the figure.

b. Apply the scale factors.

This implies that if the rotation is 90°, the X scale factor will be applied to the values of the coordinates which were originally Y-coordinates before the figure was rotated.

It is permissible to include Q-calls within definitions; the condition may exist where a definition contains a call to another definition, which in turn calls still another definition, etc. A Q-call calling a definition is defined as a Nesting Level of 1. If the definition calls another definition through a Q-call, the Nesting Level becomes 2. The maximum Nesting Level permitted is 10.

Whenever a DFL file is being processed and a Q-call is reached, the current mask level, line width, rotation and scale factors are stored before the Q-call is processed. Then the (new) rotation and scale factors, if any, are calculated and the called definition processed. The definition may change the mask level and width. When the end of the definition is reached and control returned to the point in the file where the definition was called, the previous values of mask level, line width, rotation and scale factors (if any) are restored.

11. <u>Library Update Segment</u>

Key: U

Span: N+2, N = # definitions to be deleted

Format:

U	0	0	SPAN
FLAG	0	N= #Definitionsdeleted	
NAME 1			
NAME 2			

Interpretation: The U-segment signals those systems maintaining libraries that a library update is occurring. The U-segment contains the names (i. e., numbers) of definitions to be deleted. The second half of the second word of the segment contains the number of definitions which immediately follow the U-segment which are to be added to the library. The FLAG byte is the EBCDIC character I ($=C9_{16}$) if the particular update run is to completely initialize the library; if the FLAG is EBCDIC 0 ($=D6_{16}$), then an existing library is to be updated.

C. OUTDATED DFL CHARACTERISTICS

Several older DFL files may contain some valid but no longer supported characteristics.

1. G-Segment

Previously used to specify artwork generation for Mann Artwork Program. Superseded by non-DFL program control cards.

2. D-Segment wtih Definition Length

At one time the Z key was not used to end a definition; the number of words included in the definition was placed in the fifth word of the D-segment.

D. SUMMARY: DFL LIMITS

<u>Limits</u>

P	Polygon	127 corners
O	Polygon	254 corners - all right angles, sides parallel to axes

L	Centerline	127 connected points - line may cross itself
M	Mask Level	0-23 only can be processed

Coordinates - minimum 10^{-5} inch, maximum =21.4748347 inches

Holes and Exteriors: Up to 200 holes max per exterior, but no more than 500 corners

No hole may intersect exterior or another hole.

Every hole must be inside an exterior.

No exterior may be inside a hole.

Exteriors may intersect; holes in common will be lost.

Definition Names:

1-199	Defined and called within file
200-9999	User library
10,000-32,767	System library

A definition may contain Q calls, but may not contain another definition.

A design file may not nest Q-calls up to a depth of 10.

Mask Level, Width, Rotation and Scale Factor are restored following a Q-call.

Rotations: 0

1

2

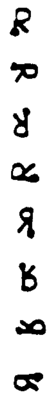
3

4

5

6

7



Mann Pattern Generator:

(measurements in inches on the reticle):

Aperture opening 0.0005 inch to 0.1200 inch steps of 0.0005 inch

Aperture angle 0° to 89°, steps of 1°

Travel 0 to 4 inches, steps of 0.00025.